Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Survey Track

Learning for Graph Matching and Related Combinatorial Optimization Problems

Junchi Yan[1*], **Shuang Yang**[2] and **Edwin Hancock**[3]
[1] Department of CSE, MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
[2] Ant Financial Services Group
[3] Department of Computer Science, University of York
yanjunchi@sjtu.edu.cn, shuang.yang@antfin.com, edwin.hancock@york.ac.uk

Abstract

This survey gives a selective review of recent development of machine learning (ML) for combinatorial optimization (CO), especially for graph matching. The synergy of these two well-developed areas (ML and CO) can potentially give transformative change to artificial intelligence, whose foundation relates to these two building blocks. For its representativeness and wide-applicability, this paper is more focused on the problem of weighted graph matching, especially from the learning perspective. For graph matching, we show that many learning techniques e.g. convolutional neural networks, graph neural networks, reinforcement learning can be effectively incorporated in the paradigm for extracting the node features, graph structure features, and even the matching engine. We further present outlook for the new settings for learning graph matching, and direction towards more integrated combinatorial optimization solvers with prediction models, and also the mutual embrace of traditional solver and machine learning components.

1 Introduction

Combinatorial optimization (CO) has been an established and indispensable research direction, which spans rich classic algorithmic solvers covering constraint satisfaction problems, integer programming, graph algorithms etc. Meanwhile, the recent decade has witnessed the surge of deep learning (DL), which allows for learnable feature extraction by cascading differentiable layers in flexible architectures. These two areas have both made profound impact from fundamental research to real-world applications, while the paths of these two areas have rarely crossed until recent years. This paper generally discusses the emerging trend for machine learning (ML), and especially dive into the area of learning graph matching, for the hope of better efficiency, scalability and even improved accuracy, with less reliance on expertise.

CO has direct business and social impact in wide applications ranging from supply chain planning to locomotive dispatching, with commercial toolboxes e.g. CPLEX and Gurobi. Yet they still suffer tractability and scalability issues. Aside from its wide existence in applications, CO also naturally raises in ML with various forms e.g. graph matching, Markov random field, conditional random field with downstream applications like pose estimation, image segmentation, entity recognition. While the computational issues can be further pronounced for more effective solution, e.g. considering the dense pixels for segmentation and dense correspondences between views.

Many combinatorial problems are NP-hard if not even harder, or practically intractable for large-scale settings for exact solution. Though still in its early phase, in our opinion, introducing learning for cost-effective (near-)optimal solutions finding has some potential advantages:

i) Data-driven and generic approximation: traditional solvers are based on experts' theoretical or/and empirical knowledge. The components can be labor-intensive and case dependent. In contrast, learning methods are expected to be free from the experts by training (with generic approximation) often in an end-to-end fashion, which allows to model real world problems in a flexible way. For instance, many combinatorial problems are based on graph structure [Bengio *et al.*, 2018], which can be readily modeled by existing graph embedding or network embedding techniques, which embed the graph information into continuous node representation. Then the constraints can also be accounted by different techniques out of which reinforcement learning can be a flexible and general approach [Khalil *et al.*, 2017].

ii) Adaptivity and reusability: learned models (or in other forms e.g. meta policy, meta rewards) are known can be transferred to relevant tasks in different ways, hence a trained combinatorial solver or a meta solver may be adapted to new tasks. It may further enhance the practical applicability by addressing the cold start problem using traditional solvers, for instance by imitation learning. In general, a specific solver could be more cost-effective than a general-purpose method, and ML can provide a generic way of building such solvers with training data rather than human knowledge. The learning can be possibly further eased by flexibly designed neural network architectures with end-to-end learning paradigm (also potential to network architecture search as there is vast literature therein).

iii) Computational efficiency: deep networks often in-

*Corresponding author is Junchi Yan. This work is partly supported by National Key Research and Development Program of China 2018AAA0100704, and NSFC 61972250, U19B203.

volve matrix multiplication and convolution. It allows highly parallel computation and full usage of GPU, especially when the discrete combinatorial problems can be mostly relaxed into the continuous domain as successfully fulfilled in some recent graph matching networks [Wang *et al.*, 2019b; Yu *et al.*, 2020]. This also holds for the graph tasks like Minimum Vertex Cover, Maximum Cut and Traveling Salesman problems as addressed in [Khalil *et al.*, 2017] by graph embedding. Then it can be accelerated by parallel computations in GPU. Another example can be found in [Bertsimas and Stellato, 2019] for solving online mixed-integer optimization problems at very high speed using neural network and a linear system based solver. In contrast, traditional solvers, which are mostly performed on CPU, often require iterative computation and more logic operation, being less parallel-friendly, more time-consuming and power intensive. Though currently traditional learning-free solvers are often more accurate.

A rich body of learning techniques have been applied to solve the combinatorial problems, though the performance of learning based solvers in general may still fall behind the well-designated classic competitors, especially on real-world complex problems and datasets. In the following, we give examples for recent advancement whereby various learning based techniques are applied, including graph neural networks [Gasse *et al.*, 2019], reinforcement learning [Bello *et al.*, 2016], multi-agent [Sghir *et al.*, 2018], etc. Specifically, the work [Huang *et al.*, 2019] solves the well known NP-hard problem for coloring very large graphs with deep reinforcement learning, and the Travelling Salesman Problem (TSP) is studied in [Vinyals *et al.*, 2015] with the proposed pointer network. Preliminary success on learning for the NP-complete decision-variant of TSP is recently attained in [Prates *et al.*, 2019]. Meanwhile, deep learning for node set is also explored in [Zaheer *et al.*, 2017] where permutation invariant objective functions are learned for node sets. For resource management, imitation learning is adopted in [Shen *et al.*, 2019] to learn the pruning policy in the optimal branch-and-bound algorithm. Interestingly their method does not resort to an end-to-end learning paradigm, and in fact can more effectively exploit the algorithm structure with few training samples. The technique for learning to branch has also been well develop in [Gasse *et al.*, 2019] by imitation learning with expert rules.

Despite the above preliminary and scattered results, learning e.g. graph attention network (GAT) in [Kool *et al.*, 2018] has shown its promising universal applicability to various combinatorial optimization problems, including TSP, Vehicle Routing Problem (VRP), Orienteering Problem (OP), Prize Collecting TSP (PCTSP) and Stochastic PCTSP (SPCTSP). Such a generality may come from GAT's ability to learn the relations between nodes and edges, especially the changeable relations among adjacent nodes, which is key to graph based combinatorial optimization problems.

Among the combinatorial problems, graph matching or namely quadratic assignment, has been a fundamental problem across different areas ranging from pattern recognition, computer vision to resource management. In fact, learning based methods are relatively few [Cho *et al.*, 2013; Caetano *et al.*, 2009; Leordeanu *et al.*, 2011] to this problem until the recent pioneering work [Nowak *et al.*, 2018; Zanfir and Sminchisescu, 2018] which boosts its performance notably. Differing from the general introduction [Bengio *et al.*, 2018; Lombardi and Milano, 2018] on learning for CO, our discussion will be focused on GM for its representativeness, hardness, generality, and usefulness. The detailed study is aimed to show how learning techniques and paradigms are disruptively transforming combinatorial optimization. The hope is making this article more self-contained, comprehensible, and hands on. Finally we also give exploratory outlook to highlight a list of (but not exhaustive) promising directions in graph matching and beyond, by the use of ML within CO.

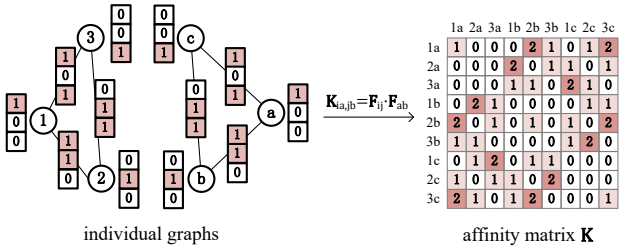

Figure 1: Illustration of the widely used affinity matrix $\mathbf{K}$ (see Eq. 1) in two-graph matching problem: graph (1, 2, 3) to graph (a, b, c). Each node or edge is represented with a three-dimensional feature vector, and the affinity is calculated by the inner product. In learning base methods, the features can be learned by CNN or GNN.

2 Towards Learning of Graph Matching

Overview. For generality, we discuss graphs with weighted edges and labeled nodes. The matching refers to establish the one-to-one node correspondence over two or more graphs, to maximize the affinity score (denoted by J – see Eq. 1) between matched nodes (with outliers), which is NP-complete. This form is more general than graph isomorphism or subgraph isomorphism, with wide connection to other problems.

As shown in Table 1, learning GM often involves i) learning to extract more tailored node features (by e.g. CNN); ii) geometric features of graph (by e.g. GNN); iii) affinity model, which are guided by different loss functions. Node assignment module is also learned in RL based methods [Liu, 2018], and extension to hypergraph and multi-graph matching solvers has also been devised [Wang *et al.*, 2019b] (see Fig. 2). In our analysis, the adoption of the above techniques in GM has also well represented their roles in CO.

2.1 Learning-free Graph Matching

We refer the traditional optimization methods for GM as classic ones, to differentiate from the learning-based methods.

Classic Two-Graph Matching

Objective formulation. Two-graph matching can be modeled as the quadratic assignment problem (QAP), which is known NP-complete. Its most general form called Lawler's QAP has been widely adopted in literature [Leordeanu and Hebert, 2005; Leordeanu *et al.*, 2009; Cho *et al.*, 2010; Zhang *et al.*, 2019]:

$$J(\mathbf{x}) = \mathbf{x}^\top \mathbf{K}\mathbf{x}, \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^\top \mathbf{1}_{n_1} \leq \mathbf{1}_{n_2}, \mathbf{X} \in \{0,1\}^{n_1 \times n_2} \quad (1)$$

where $\mathbf{x} = \text{vec}(\mathbf{X})$ is column-wise vectorized version of matrix $\mathbf{X} \in \{0,1\}^{n_1 \times n_2}$ which denotes the binary correspondence matrix (namely assignment matrix). Note $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is the so-called affinity matrix, whereby the diagonal elements store the vertex-to-vertex similarity while off-diagonal carry the edge-to-edge similarity. An illustration of $\mathbf{K}$ is presented in Fig. 1. Another popular form is Koopmans-Beckmann's (KB) QAP [Loiola *et al.*, 2007], as a special case for Lawler's QAP by setting $\mathbf{K} = \mathbf{F}_2 \otimes \mathbf{F}_1$, where $\mathbf{F}$ are the weighted adjacency matrices:

$$J(\mathbf{X}) = \underbrace{\text{tr}(\mathbf{X}^\top \mathbf{F}_1 \mathbf{X} \mathbf{F}_2)}_{\text{edge affinity}} + \underbrace{\text{tr}(\mathbf{K}_p^\top \mathbf{X})}_{\text{node affinity}} \tag{2}$$

Note the values of the node-to-node affinity matrix $\mathbf{K}_p$ (similarly for $\mathbf{F}$) depend on applications. In computer vision, SIFT or CNN features are used, while in social network analysis, node feature can be extracted by user's behavior. To measure the value in $\mathbf{K}$, usually Gaussian kernel is used for real-value data like image features, while it can be edit distance between texts e.g. username in social networks. A parametric form of $\mathbf{K}$ can also be learned as summarized in Table 1.

The above formulas only consider the second-order affinity, and the higher-order similarity among a pair of hyperedge (i.e. vertex tuple) can be readily encoded by affinity tensor. This leads a popular objective [Lee *et al.*, 2011] as follows for maximization (constraints are omitted for briefly):

$$J(\mathbf{x}) = \mathbf{H} \otimes_1 \mathbf{x} \cdots \otimes_p \mathbf{x} \tag{3}$$

where $\mathbf{H}$ is the affinity tensor and p is the order of the affinity. Termed as hypergraph matching, various higher-order methods [Zass and Shashua, 2008; Chertok and Keller, 2010; Duchenne *et al.*, 2011; Yan *et al.*, 2015b; Ngoc *et al.*, 2015] have been proposed for improved matching accuracy at the cost of increased time and space complexity. The common strategy is to transform the higher-order problem into the second-order case in an iterative fashion.

Learning-free optimization. Some methods search the solution directly in assignment matrix space via heuristics. [Leordeanu *et al.*, 2009] is devised with the hope that an optimal solution can be found along a (quasi) discrete solution course. The methods [Lee *et al.*, 2010; Suh *et al.*, 2012] generate discrete solutions via Monte Carlo Sampling. [Adamczewski *et al.*, 2015] devises a tailored Tabu search for graph matching. The idea of gradient descent with projection to constraint is also explored in [Yan *et al.*, 2015b].

Meanwhile, relaxation techniques have been devised, by which the continuation method (i.e. path-following) is often adopted. In [Gold and Rangarajan, 1996], a deterministic annealing procedure is performed in the continuous space. Similar path-following techniques are widely used in recent work [Zhou and Torre, 2016].

We briefly discuss three representative ways as follows: i) spectral relaxations on the matching matrix [Leordeanu and Hebert, 2005; Cour *et al.*, 2006]. The matching constraint is loosened by $\|\mathbf{x}\|_2 = 1$ which can be solved efficiently; ii) doubly-stochastic (DS) relaxation on the matching matrix [Gold and Rangarajan, 1996; Leordeanu *et al.*, 2009]. Note DS matrix is the convex-hull of the assignment matrix; iii) semidefinite-programming (SDP) [Torr, 2003; Schellewald and Schnörr, 2005]. The relaxation model in [Schellewald and Schnörr, 2005] can be written as:

$$\min_{\mathbf{Y}} \text{Tr}(\mathbf{Q}\mathbf{Y}) \quad \text{s.t.} \quad \mathbf{Y} \succeq \mathbf{0}, \quad \text{Tr}(\mathbf{A}_i \mathbf{Y}) = \mathbf{c}_i \tag{4}$$

where constraints are defined by a series of $\mathbf{A}_i$ and $\mathbf{c}_i$. There is off-the-shelf solver for SDP problem, while the derived variable $\mathbf{Y} \in \mathbb{R}^{(n_1 n_2+1)\times(n_1 n_2+1)}$ causes scalability issue.

Multiple-Graph Matching

There are emerging line of works on joint matching of multiple graphs, which bears several merits compared with two-graph case. First, the graphs are often given in batch and a cycle-consistent matching across graphs is welcomed. Here consistency refers to the fact that the correct correspondence shall lead to the loop node mapping over three or more graphs, for instance $\mathbf{X}_{12} = \mathbf{X}_{13}\mathbf{X}_{32}$. However, performing two-graph matching pair-wisely cannot guarantee such a consistency. Moreover, two-graph matching can be inherently ill-posed because: i) the graphs are often noisy, ii) modeling the affinity objective is nontrivial. These factors can both incur the mathematically global solution cannot guarantee a meaningful perfect matching due to the deviation in objective. This partly motivates the idea of learning for graph matching in the sense of more meaningful objective design.

Here we introduce the learning-free multi-graph matching methods and discuss their pros and cons. We divide recent multi-graph matching methods into three groups:

i) Methods in the first group transform the multi-graph matching problem into a pairwise matching task at each iteration. Hence off-the-shelf two-graph matching solvers can be readily reused.

ii) While the second group involves methods that search matching composition chain based on initial (noisy) pairwise matching result to optimize both affinity and cycle-consistent.

iii) The methods of last group take a clustering or low rank recovery perspective in feature space.

The first category of methods often start with a basic objective for joint matching N graphs (constraint omitted), to maximize the overall affinity score J in the following quadratic form, where $\mathbf{K}_{ij}$ is the affinity matrix between two graphs:

$$\mathbb{X}^* = \arg\max_{\mathbb{X}} \sum_{i,j=1,i\neq j}^{N} \mathbf{x}_{ij}^\top \mathbf{K}_{ij} \mathbf{x}_{ij} \tag{5}$$

By introducing a series of basis $\{\mathbf{X}_{rk}\}_{k=1,k\neq r}^{N}$ regarding with a pre-selected reference graph $\mathcal{G}_r$, [Yan *et al.*, 2015a] shows that the above problem can be iteratively solved by transforming it into a QAP problem in each iteration. The iteration is conceptually performed on a super-graph regarding each graph as its vertex, and $\mathcal{G}_r$ serves as the central vertex for iterative updating. Such a star-shape centralized framework inherently suffers from fragility as the information flow are all through $\mathcal{G}_r$ which becomes the bottleneck and the matching error can also be accumulated. The underlying assumption is that $\mathcal{G}_r$ shall be similar or easier to match with other graphs, which however can not hold in many cases.

To mitigate the above issue, a distributed framework is proposed in [Yan *et al.*, 2016]. Similar to the star-shape centralized framework, a matching will be updated iteratively that involves an intermediate graph. The difference is that there

method	CNN	GNN module	affinity metric	loss function
[Nowak *et al.*, 2018]	none	GCN	inner-product	multi-class cross-entropy
[Zanfir and Sminchisescu, 2018]	VGG16	none	weighted exponential	pixel offset regression
[Wang *et al.*, 2019a]	VGG16	GCN + cross-graph conv.	weighted exponential	binary cross-entropy (BCE)
[Jiang *et al.*, 2019a]	VGG16	Laplacian smoothing/sharpening + cross-graph convolution	weighted exponential	binary cross-entropy + sparsity norm
[Yu *et al.*, 2020]	VGG16	Channel independent embedding + cross-graph convolution	weighted exponential	binary cross-entropy with Hungarian attention
[Fey *et al.*, 2020]	VGG16	SplineCNN/GIN/cross-GNN	inner-product	BCE+ neighbor consensus

Table 1: Three main modules for learning: CNN, GNN, affinity model. CNN (VGG16) and GNN (different variants) is widely used for extracting visual features to build the graph attributes and structure, respectively. The affinity can also be learned effectively by inner-product or weighted exponential form, while cross-entropy is often used as loss, based on a double-stochastic matrix converted by Skinhorn net.

is no centralized reference graph, and the intermediate graph is distributed among every graph in the set. Moreover, the consistency constraint is relaxed by adding a consistency regularizer C_p on affinity J for more robust optimization:

$$k^* = \arg\max_{k=1}^{N} \underbrace{(1-\lambda)J(\mathbf{X}_{ik}\mathbf{X}_{kj})}_{\text{affinity score}} + \underbrace{\lambda C_p(\mathbf{X}_{ik}\mathbf{X}_{kj}, \mathbb{X})}_{\text{consistency score}} \quad (6)$$

This approach estimates matchings between dissimilar graphs by composition along a path of similar graphs. The cycle consistency metric has been recently more efficiently explored in [Guibas *et al.*, 2019] that only samples a subset of weighted cycles according theoretical study.

Another body of work aim to recover a globally consistent pairwise matching set from putative pairwise matchings. Spectral techniques [Kim *et al.*, 2012; Pachauri *et al.*, 2013; Huang and Guibas, 2013] are developed to extract the consistent matches by the spectrum (top eigenvectors) of the matrix composed of all putative matches and these early works often assume bijection among all the graphs. The underlying rationale is that the problem can be formulated as quadratic integer programming which can be relaxed into a generalized Rayleigh problem [Pachauri *et al.*, 2013].

In the seminal work [Huang and Guibas, 2013], the authors show theoretical conditions for exact recovery. They note that if the pairwise matchings are cycle-consistent then the bulk matrix storing all matchings is low-rank and positive semidefinite. This leads to a convex relaxation method for estimating cycle-consistent matchings by finding the nearest positive semidefinite matrix to the input matrix stacking by all initial matchings. Improvement is made in [Chen *et al.*, 2014] by assuming the underlying rank of variable matrix can be estimated reliably. Then two extensions are made which are further improved in [Zhou *et al.*, 2015]: i) partial matching to allow for when different groups of inliers are shared among different graph subsets; ii) robust recovery from a small fraction of observation given large portion of hidden or erroneous matches. Another typical setting is to identify the inliers commonly shared by every graph, as addressed in [Wang *et al.*, 2018] which requires additional constraint from motion estimation. Finally there are also recent works on distributed methods for both memory and time efficiency. [Leonardos *et al.*, 2017] proposes a decentralized version of the centralized spectral method [Pachauri *et al.*, 2013]. Akin to [Pachauri *et al.*, 2013], their method can only be applied to the bijection case, which is addressed in [Hu *et al.*, 2018] by a theoretical study on the connection between the cycle-consistency on overlapped clusters, and that on the whole graph set.

2.2 Shallow Learning for Graph Matching

In all the above methods, the objective is pre-given, either for the node and edge attributes (for KB's QAP), or the direct pairwise affinity between nodes and edges (for Lawler's QAP). While it is often the case that the objective can be better modeled via learning, instead of a handcrafted loss. For instance in image matching, CNN features pretrained on ImageNet may be more discriminative than SIFT feature for matching. The efficacy can be further boosted by end-to-end training the CNN tailored to the matching task. This is understandable since the raw CNN features that are good at object recognition may account more for the intra-class variability which loses the sensitivity to local feature changes.

Structured Learning-based Methods

Structured SVM has been adopted in [Cho *et al.*, 2013] for a unified framework for learning the weight of each node and edge, instead of a homogeneous weight allocation in learning-free methods. Specifically, the sense of structure learning refers to the prediction of graph edge weights whereby the graphs are assumed of equal size and the vertex are fully connected. This approach incorporates a series of previous GM learning [Caetano *et al.*, 2009; Leordeanu *et al.*, 2012; Torresani *et al.*, 2008] as its special cases. The weights can also be learned in an unsupervised [Leordeanu *et al.*, 2012] or semi-supervised [Leordeanu *et al.*, 2011] (for hypergraph) fashion. Compared the following deep network based approaches, these methods have limited model capacity which can mostly only account for the different affinity weights of nodes and edges for matching. While the node-wise feature representation and structural information cannot be learned.

Network Embedding-based Learning

Large-scale GM, often in the context of network alignment, has also evolved into a well-established research area. For scalability issue, shallow network embedding has been the dominant tools instead of costive deep learning methods.

Early works mainly rely on handcrafted features. Given seed alignments, MNA [Kong *et al.*, 2013] extracts pairwise features including number of common neighbors, Jaccard similarity of users' neighborhoods, and Adamic/Adar measure from networks and then correspondences are found by stable matching. BASS [Cao and Yu, 2016] jointly models consistencies for handcrafted features, including Jaccard similarity of users' neighborhoods, user name edit distance, and the features of social contents, and then the matching classifier is trained in EM by bootstrapping. A Markov chain is

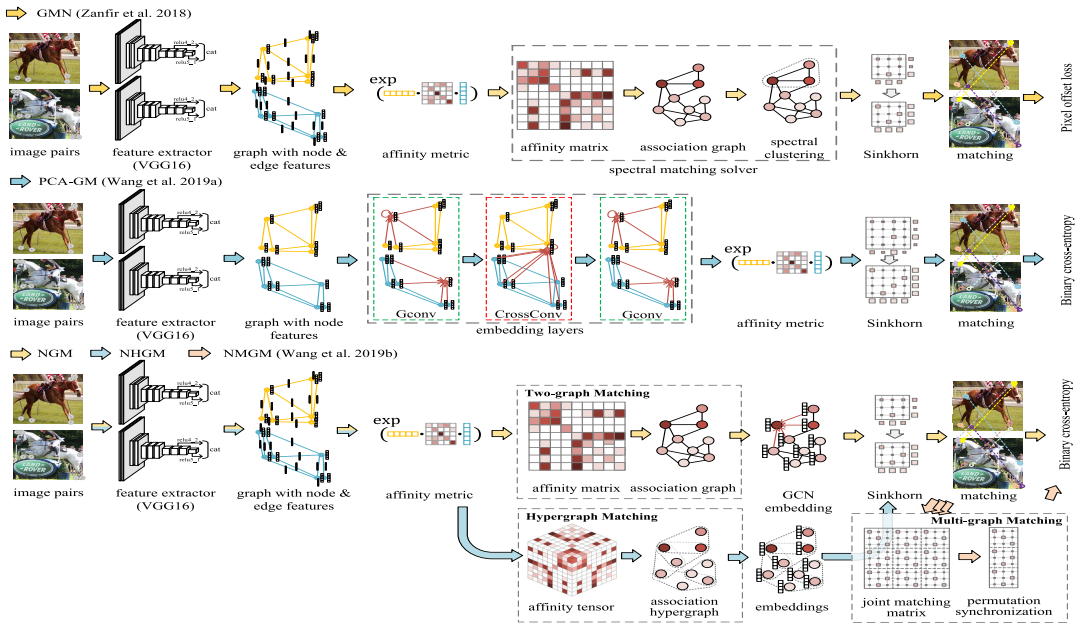

Figure 2: Illustration of recent GM learning methods (see details in Table 1). Compared to [Zanfir and Sminchisescu, 2018] that only uses CNN models to extract image features, GNN is used in [Wang *et al.*, 2019a] for jointly embedding the two input graphs for graph structure learning. Hypergraph and multiple GM learning are further devised in [Wang *et al.*, 2019b] by association graph embedding.

modeled [Zhang *et al.*, 2015] to gradually recover the missing matchings from seeds. These methods, in general, lack the capacity to fully explore the structure using learning.

Recently learning based, especially embedding based methods become more popular, which mostly follow a weakly-supervised setting with seed alignments for expansion. The basic idea in [Tan *et al.*, 2014; Liu *et al.*, 2016; Li *et al.*, 2016] is to train node embeddings by forcing the seed nodes across networks to be the same. [Du and Tong, 2019] shows multi-resolution embedding on multiple networks brings further improvement on network alignment. [Wang *et al.*, 2019c] proposes a reinforcement learning approach on bipartite graph matching. There are works [Man *et al.*, 2016; Zhou *et al.*, 2018] follow a simple pipeline starting with embedding: i) generate each node's embedding based on matrix factorization [Tan *et al.*, 2014] or Skip-gram [Man *et al.*, 2016], supervised by seeds; ii) measure pairwise embedding similarity between nodes by e.g. cosine [Man *et al.*, 2016; Zhou *et al.*, 2018] or Euclidean distance [Heimann *et al.*, 2018]; iii) match node pairs with high similarity.

There are recently fully unsupervised network alignment methods. In [Du *et al.*, 2019], the authors propose a cross-graph random walk based embedding technique for alignment without any seed alignment. [Xu *et al.*, 2019] jointly learns cross-network embedding and performs matching by adopting Gromov-Wasserstein learning which is the second-order generalization form of Sinkhorn, and its formulation can also be regarded as a special case of Lawler's QAP.

2.3 Deep Neural Network-based Learning

In computer vision, the seminal work [Zanfir and Sminchisescu, 2018] firstly adopts CNN for feature extraction to build the affinity matrix $\mathbf{K}$, followed by a differentiable (but fixed) spectral solver [Leordeanu and Hebert, 2005] to approximately obtain the matching. Note in this network, only image features can be learned via CNN while the structure information is still hard to model and in fact poses persistent challenge as in learning-free methods. To solve this issue, graph neural network (GNN) has become a important way to help embed node structure information into graph node representation, especially for graphs of moderate size.

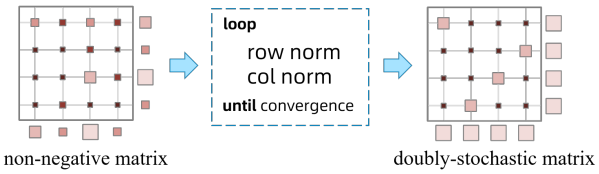

Figure 3: Demonstration of the Sinkhorn operation as performed in the Sinkhorn network, which involves alternative row-wise and column-wise normalization until converging to a doubly-stochastic matrix or for certain rounds. To obtain the final matching matrix, Hungarian method is adopted to convert the doubly-stochastic matrix into a binary permutation one. This operation is key to graph matching networks that bridge the gap between continuous domain which is more friendly to deep learning, to the final discrete solution.

The work [Nowak *et al.*, 2018] considers alignment between two identical graphs, with one graph distorted by noise. GNN is used to extract node's structural features and Sinkhorn network [Adams and Zemel, 2011] is adopted as a differentiable and fixed layer [Kosowsky and Yuille, 1994] to solve the resulting linear assignment problem. The Sinkhorn network has served as a building block in many combinatorial learning pipelines by repeatedly performing row and column-normalization until convergence. As shown in Fig. 3, Sinkhorn algorithm turns a non-negative input matrix into a

doubly-stochastic one, as the convex hull of the final discrete permutation matrix, which can be converted by the Hungarian method [Kuhn, 1955]. The recent work [Wang *et al.*, 2019a] combines the CNN and (cross-graph) GNN to extract the local image features and the geometry information by the formed graph, and a so-called permutation loss is devised by the integration of the Sinkhorn layer and cross-entropy loss, in contrast to the inner-product [Nowak *et al.*, 2018] or regression loss [Zanfir and Sminchisescu, 2018].

This idea is further improved in [Wang *et al.*, 2019b; Yu *et al.*, 2020]: the former work proposes to directly perform embedding on the association graph for matching, which corresponds to the most general Lawler's QAP, instead of the KB's form in previous works. Also multi-graph matching and hypergraph matching by edge embedding are both fulfilled as an extension to [Wang *et al.*, 2019a]. In [Yu *et al.*, 2020], a Hungarian attention mechanism is devised to solve the overfitting issue in permutation loss. Hungarian attention computes a discrete matching result by Hungarian algorithm, paying more attention on mismatched nodes and less on correct ones. They also develop a channel-independent edge embedding technique to enhance graph feature extraction. [Li *et al.*, 2019] fuses information across graphs with attention mechanism on GNN based embedding to measure the similarity between graphs. [Fey *et al.*, 2020] further proposes a graduated refinement approach to ensure the local consensus between graphs, in a salable way. Refer to Fig. 2 and Table 1 for illustration and comparison.

Finally, it shall be noted that it is the GNN (e.g. GCN) rather than CNN that provides the capability for embedding the graph structure into vectors in continuous space. Such continuous embedding can either be used in graph structure feature extraction [Wang *et al.*, 2019a] or direct assignment solution scoring that is performed on the association affinity matrix [Wang *et al.*, 2019b].

2.4 On Reinforcement Learning for GM

Using reinforcement learning (RL) for GM is still in its infancy. The preliminary work [Liu, 2018] takes a progressive matching strategy by matching one pair of nodes each time, which naturally fits with the sequential decision paradigm in RL. As an early work in RL for GM, its problem setting is relatively simple: seeking the correspondence between two graphs with no edge nor node attribute. Hence the designated reward is also simple: accounting for the binary structural consistency between a pair of two nodes without knowing the ground truth matching. Deep Q-learning is adopted for training the neural Q-function whose input is multi-layer node embedding by GNN in an unsupervised manner by using the above reward function. The hope is that the learned Q-table and policy can well fit the specific graph data instead of traditional heuristics. Experiments involve only synthetic data with no testing on real-world data as the approach is relatively elementary. So far there is still little work using RL for GM [Wang *et al.*, 2019d], which is worth further study.

2.5 Joint Graph Matching with Other Tasks

GM can also be jointly solved with other tasks, which can be of practical importance. The tasks of social network alignment and link prediction can benefit each other and this idea has been explored in [Du *et al.*, 2019] via cross-network embedding. In vision, the single image may contain repeated objects/structures, and it is welcomed to solve graph cut and matching in one-shot [Yu *et al.*, 2018]. In practical scenarios, there can be different graphs hence clustering is needed for meaningful matching within each cluster. Clustering-aware multiple graph matching is developed [Wang *et al.*, 2020]. In the above latter two settings, cuts and clustering are also typical combinatorial problems, which pose further challenge for a joint solution. We expect learning will be more frequently used for solving more joint optimization tasks.

3 Conclusive Remarks for Graph Matching

Based on the above description, we try to give some conclusive remarks on the transformation from classic methods to learning-based ones in the following aspects:

3.1 What to Learn?

Node-wise representation. Different from traditional GM focusing on the combinatorial matching itself, learning can provide more broad space to enhance the overall matching pipeline. For visual object matching, image features can be better extracted by learning CNN tailored to the matching task which is ignored in the classic GM setting.

Edge-wise representation. For the established graphs, its structure can also be effectively extracted by learning rather than a fixed embedding procedure. For large-scale network, shallow models e.g. network embedding methods like DeepWalk [Perozzi *et al.*, 2014], struc2vec [Ribeiro *et al.*, 2017] are more popular than the GNN model, which is often used in moderate size problem. The basic idea is to embed the geometric structure around the node into a unary embedding, as such the NP-complete GM is reduced into a much easier linear assignment problem.

Matching solver. In the previous learning methods [Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019a], the learning is focused on the above feature representation part. While in the more recent work [Wang *et al.*, 2019b], it has been successfully directly applied in decision part, i.e. scoring the assignment solution via embedding on the so-called association graph whose weighted adjacency matrix is the affinity matrix in two-graph matching (see the bottom row of Fig. 2).

3.2 How to Learn?

The pipeline based on GNN with Sinkhorn net and cross-entropy loss has been well developed and widely used (see Table 1. In contrast, alternative methods receive less attention and still have not achieved comparable performance e.g. RL approaches. In our analysis, the reasons are as follows: GNN helps transform the problem into a linear assignment task with embedding that can encode both second (or higher)-order information, which otherwise need careful and costive treatment in traditional methods [Yan *et al.*, 2015b]. The essential role for GNN in combinatorial problems has been advocated in literature [Khalil *et al.*, 2017]. Note that after Sinkhorn re-normalization, the output becomes a doubly-stochastic matrix which can be regarded as a multi-class dis-

tribution hence the cross-entropy loss is a natural choice as loss given ground truth. In this sense, the differentiable Sinkhorn net plays a vital role in connecting the continuous computing which is more friendly to GPUs and the decision variables i.e. assignment matrix.

3.3 What is Next?

We believe reinforcement learning has promising potential in the more challenging multiple graph matching problems, seeing its recent success in multi-object tracking, either by deep networks [Ren *et al.*, 2018], multi-agent [Rosello and Kochenderfer, 2018] or both [Jiang *et al.*, 2019b] to address a similar data association problem. In particular, we would like to emphasize that the two-graph matching setting in fact has been largely solved by existing Sinkhorn net based learning methods, which involves turning the quadratic assignment problem into the more easier linear assignment problem by node embedding. Hence one may pay more attention to the multi-graph case which is also very common in practice.

To our best knowledge, the only existing multi-graph matching learning method [Wang *et al.*, 2019b] is based on GNN embedding, which suffers accuracy loss in feed-forward computing as it uses the approximate spectral matching to convert input two-graph matchings into global consistent ones for final loss computing. Moreover, the spectral layer is fixed and non-learnable, which limits the model capacity. How to devise more principled multi-graph matching learning methods is challenging, and RL as well as multi-agent reinforcement learning, may play a more important role compared with that in two-graph matching.

Moreover, meta learning [Finn *et al.*, 2017] can also be of great potential for graph matching, as well as the general QAP problems. It has shown in [Wang *et al.*, 2019a] that the learned model based on the KB's form (especially the CNN features) have some transferability across different categories of objects. For more general Lawler's QAP, one way to enable such a transfer learning protocol might be in [Wang *et al.*, 2019b] directly taking the association graph induced affinity matrix for embedding learning. Here the GNN embedding differs fundamentally from the embedding on the input graph, and the embedding model can be used to transfer across different QAP tasks.

4 Further Discussion on CO and Outlook

Finally we discuss learning for CO in some interesting directions: i) integration with traditional combinatorial solvers, ii) integration with predictive models (a.k.a. stochastic combinatorial optimization) and iii) developing reinforcement learning for CO. The first refers to the idea of how to reuse traditional solvers, especially for few-sample cases. The second has practical impact for streamlining whole analytics systems that involve both prediction and decision making. While we believe the third one is a general methodology to explore the complex problem structure in CO.

Fusing ML with traditional CO solvers. Although many learning based works are shown above, the gap between neural network and CO is still huge. On the one hand, directly applying neural networks may produce unsatisfactory results in many CO problems as their discrete nature pose particular challenge to networks computed in continuous domain. On the other hand, CO problems are strongly based on high-level mathematical abstraction on real-world problems, with which traditional solvers also suffer from the rigidity and expert rules/heuristics, which has difficulty in modeling the physical world in a generic way. Fusing traditional solvers and learning based methods seems an promising way to bridge complex real world to the highly-abstracted CO problems, and to combine the best of the two areas.

Neural network can be combined with, or integrated in, an algorithmic solver to more effectively explore the algorithmic structure. The traditional solver is also a good teacher for neural networks by certain means e.g. imitation learning, especially given few samples for training [Gasse *et al.*, 2019]. Also, it is beneficial to incorporate traditional solvers into a pipeline with flexibly placed neural networks for the modeling of real-world problems. Recent work [Vlastelica *et al.*, 2020] has shown this can be fulfilled with differentiable end-to-end learning by adopting deep networks for problem modeling and traditional solvers to the modeled problem.

Fusing predictive model with CO. Many prediction modules for real-world problems, especially deep neural network based models often issue deterministic point-level estimation (e.g. regression or descritized classification output), and the fixed estimation (without uncertainty interval) is then fed into a CO engine for decision making. In fact, these two parts are historically often separately devised. However, uncertainty is ubiquitous in real-world problems where traditional CO researchers have been working on the long-standing stochastic operations research for decades [Dohi *et al.*, 2007]. Moreover, with modern deep learning paradigms the continuous valued network prediction can also be viewed as a probability distribution. We believe a new learning paradigm by accounting for the uncertainty from prediction to decision is promising, corresponding to the related traditional learning-free area i.e. stochastic operations research.

Embracing RL in CO. For one of the most important research and application directions in CO, i.e. integer programming, existing learning-based algorithms e.g. [Liu *et al.*, 2020; Gasse *et al.*, 2019; Balcan *et al.*, 2018] rarely consider RL in their learning pipeline. Future research may include boosting the acceleration performance of CO solvers by RL and explore problem-specific strategies. In particular, branch-and-bound can be viewed as a Markov Decision Process [Gasse *et al.*, 2019]. In addition, learning acceleration strategies for CO also naturally rise in the meta-learning setting, where recent meta-learning advances can be explored for the generalization-ability among relevant CO tasks in combination with RL.

References

[Adamczewski *et al.*, 2015] K. Adamczewski, Y. Suh, and K. Lee. Discrete tabu search for graph matching. In *ICCV*, 2015.

[Adams and Zemel, 2011] Ryan Prescott Adams and Richard S Zemel. Ranking via sinkhorn propagation. *arXiv:1106.1925*, 2011.

[Balcan *et al.*, 2018] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *ICML*, 2018.

[Bello *et al.*, 2016] I. Bello, H. Pham, Q. V Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[Bengio *et al.*, 2018] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv preprint arXiv:1811.06128*, 2018.

[Bertsimas and Stellato, 2019] Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. *arXiv preprint arXiv:1907.02206*, 2019.

[Caetano *et al.*, 2009] T. Caetano, J. McAuley, L. Cheng, Q. Le, and A. J. Smola. Learning graph matching. *TPAMI*, 31(6):1048–1058, 2009.

[Cao and Yu, 2016] X. Cao and Y. Yu. Bass: A bootstrapping approach for aligning heterogenous social networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2016.

[Chen *et al.*, 2014] Y. Chen, L. Guibas, and Q. Huang. Near-optimal joint object matching via convex relaxation. In *ICML*, 2014.

[Chertok and Keller, 2010] M. Chertok and Y. Keller. Efficient high order matching. *TPAMI*, 2010.

[Cho *et al.*, 2010] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*, 2010.

[Cho *et al.*, 2013] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *ICCV*, 2013.

[Cour *et al.*, 2006] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006.

[Dohi *et al.*, 2007] Tadashi Dohi, Shunji Osaki, and Katsushige Sawaki. Recent advances in stochastic operations research. 2007.

[Du and Tong, 2019] B. Du and H. Tong. Mrmine: Multi-resolution multi-network embedding. In *CIKM*, 2019.

[Du *et al.*, 2019] X. Du, J. Yan, and H. Zha. Joint link prediction and network alignment via cross-graph embedding. In *IJCAI*, 2019.

[Duchenne *et al.*, 2011] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. A tensor-based algor ithm for high-order graph matching. *PAMI*, 2011.

[Fey *et al.*, 2020] M. Fey, J. E Lenssen, C. Morris, J. Masci, and N. M Kriege. Deep graph matching consensus. In *ICLR*, 2020.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th ICML-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[Gasse *et al.*, 2019] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *NeurIPS*, 2019.

[Gold and Rangarajan, 1996] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *TPAMI*, 1996.

[Guibas *et al.*, 2019] L. Guibas, Q. Huang, and Z. Liang. A condition number for joint optimization of cycle-consistent networks. In *NIPS*, 2019.

[Heimann *et al.*, 2018] M. Heimann, H. Shen, T. Safavi, and D. Koutra. Regal: Representation learning-based graph alignment. In *CIKM*, 2018.

[Hu *et al.*, 2018] N. Hu, Q. Huang, B. Thibert, and L. Guibas. Distributable consistent multi-graph matching. *CVPR*, 2018.

[Huang and Guibas, 2013] Q. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *SGP*, 2013.

[Huang *et al.*, 2019] Jiayi Huang, Mostofa Patwary, and Gregory Diamos. Coloring big graphs with alphagozero. *arXiv:1902.10162*, 2019.

[Jiang *et al.*, 2019a] B. Jiang, P. Sun, J. Tang, and B. Luo. Glmnet: Graph learning-matching networks for feature matching. *arXiv preprint arXiv:1911.07681*, 2019.

[Jiang *et al.*, 2019b] M. Jiang, Z. Pan T. Hai, H. Wang, Y. Jia, and C. Deng. Multi-agent deep reinforcement learning for multi-object tracker. *IEEE ACCESS*, 2019.

[Khalil *et al.*, 2017] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In *NIPS*, 2017.

[Kim *et al.*, 2012] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. In *SIGGRAPH*, 2012.

[Kong *et al.*, 2013] X. Kong, J. Zhang, and P. S Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, 2013.

[Kool *et al.*, 2018] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2018.

[Kosowsky and Yuille, 1994] Jeffrey J Kosowsky and Alan L Yuille. The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural networks*, 7(3):477–490, 1994.

[Kuhn, 1955] H. W. Kuhn. The hungarian method for the assignment problem. In *Export. Naval Research Logistics Quarterly*, pages 83–97, 1955.

[Lee *et al.*, 2010] J. Lee, M. Cho, and K.M. Lee. A graph matching algorithm using data-driven markov chain monte carlo sampling. In *ICPR*, 2010.

[Lee *et al.*, 2011] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted randomwalks. In *CVPR*, 2011.

[Leonardos *et al.*, 2017] S. Leonardos, X. Zhou, and K. Daniilidis. Distributed consistent data association. *ICRA*, 2017.

[Leordeanu and Hebert, 2005] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[Leordeanu *et al.*, 2009] M. Leordeanu, M. Hebert, and R Sukthankar. An integer projected fixed point method for graph matching and map inference. In *NIPS*, 2009.

[Leordeanu *et al.*, 2011] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Semi-supervised learning and optimization for hypergraph matching. In *ICCV*, 2011.

[Leordeanu *et al.*, 2012] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *Int. J. Comput. Vis.*, pages 28–45, 2012.

[Li *et al.*, 2016] L. Li, W. K. Cheun, L. Xin, and L. Liao. Aligning users across social networks using network embedding. In *IJCAI*, 2016.

[Li *et al.*, 2019] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, pages 3835–3845, 2019.

[Liu *et al.*, 2016] Li Liu, William K Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *IJCAI*, 2016.

[Liu *et al.*, 2020] Yan-li Liu, Chu-min Li, Hua Jiang, and Kun He. A learning based branch and bound for maximum common subgraph problems. In *AAAI*, 2020.

[Liu, 2018] X. Liu. Graph matching by graph nerual network. *Master Thesis, University of Illinois at Urbana-Champaign*, 2018.

[Loiola *et al.*, 2007] E. M. Loiola, N. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *EJOR*, 2007.

[Lombardi and Milano, 2018] M. Lombardi and M. Milano. Boosting combinatorial problem modeling with machine learning. In *IJCAI*, 2018.

[Man *et al.*, 2016] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng. Predict anchor links across social networks via an embedding approach. In *IJCAI*, 2016.

[Ngoc *et al.*, 2015] Q. Ngoc, A. Gautier, and M. Hein. A flexible tensor block coo rd inate ascent scheme for hypergraph matching. In *CVPR*, 2015.

[Nowak *et al.*, 2018] A. Nowak, S. Villar, A. Bandeira, and J. Bruna. Revised note on learning quadratic assignment with graph neural networks. In *DSW*, 2018.

[Pachauri *et al.*, 2013] D. Pachauri, R. Kondor, and S. Vikas. Solving the multi-way matching problem by permutation synchronization. In *NIPS*, 2013.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[Prates *et al.*, 2019] M. Prates, P. Avelar, H. Lemos, L. Lamb, and M. Vardi. Learning to solve np-complete problems: A graph neural network for decision tsp. In *AAAI*, 2019.

[Ren *et al.*, 2018] L. Ren, X. Yuan, J. Lu, M. Yang, and J. Zhou. Deep reinforcement learning with iterative shift for visual tracking. In *ECCV*, 2018.

[Ribeiro *et al.*, 2017] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM, 2017.

[Rosello and Kochenderfer, 2018] P. Rosello and M. J. Kochenderfer. Multi-agent reinforcement learning for multi-object tracking. In *AAMAS*, 2018.

[Schellewald and Schnörr, 2005] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, 2005.

[Sghir *et al.*, 2018] Ines Sghir, Ines Ben Jaafar, and Khaled Ghédira. A multi-agent based optimization method for combinatorial optimization problems. *International Journal on Artificial Intelligence Tools*, 27(05):1850021, 2018.

[Shen *et al.*, 2019] Y. Shen, Y. Shi, J. Zhang, and K. B Letaief. Lorm: Learning to optimize for resource management in wireless networks with few training samples. *IEEE Transactions on Wireless Communications*, 2019.

[Suh *et al.*, 2012] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential monte carlo. In *ECCV*, 2012.

[Tan *et al.*, 2014] S. Tan, Z. Guan, D. Cai, X. Qin, J. Bu, and C. Chen. Mapping users across networks by manifold alignment on hypergraph. In *AAAI*, 2014.

[Torr, 2003] P. H. S. Torr. Solving markov random fields using semidefinite programmin. In *AISTATS*, 2003.

[Torresani *et al.*, 2008] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, 2008.

[Vinyals *et al.*, 2015] O.l Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *NIPS*, 2015.

[Vlastelica *et al.*, 2020] M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolínek. Differentiation of blackbox combinatorial solvers. 2020.

[Wang *et al.*, 2018] Q. Wang, X. Zhou, and K. Daniilidis. Multi-image semantic matching by mining consistent features. In *CVPR*, 2018.

[Wang *et al.*, 2019a] R. Wang, J. Yan, and X. Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, 2019.

[Wang *et al.*, 2019b] R. Wang, Ju. Yan, and X. Yang. Neural graph matching network: Learning lawlers quadratic assignment problem with extension to hypergraph and multiple-graph matching. *arXiv preprint arXiv:1911.11308*, 2019.

[Wang *et al.*, 2019c] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *ICDE*, 2019.

[Wang *et al.*, 2019d] Yansheng Wang, Yongxin Tong, Cheng Long, Pan Xu, Ke Xu, and Weifeng Lv. Adaptive dynamic bipartite graph matching: A reinforcement learning approach. In *ICDE*, pages 1478–1489. IEEE, 2019.

[Wang *et al.*, 2020] T. Wang, Z. Jiang, and J. Yan. Clustering-aware multiple graph matching via decayed pairwise matching composition. In *AAAI*, 2020.

[Xu *et al.*, 2019] H. Xu, D. Luo, H. Zha, and L. Carin. Gromov-wasserstein learning for graph matching and node embedding. *ICML*, 2019.

[Yan *et al.*, 2015a] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu. Consistency-driven alternating optimization for multi-graph matching: A unified approach. *IEEE Transactions on Image Processing*, 24(3):994–1009, 2015.

[Yan *et al.*, 2015b] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. Chu. Discrete hyper-graph matching. In *CVPR*, 2015.

[Yan *et al.*, 2016] J. Yan, M. Cho, H. Zha, X. Yang, and S. Chu. Multi-graph matching via affinity optimization with graduated consistency regularization. *TPAMI*, 2016.

[Yu *et al.*, 2018] T. Yu, J. Yan, J. Zhao, and B. Li. Joint cuts and matching of partitions in one graph. In *CVPR*, 2018.

[Yu *et al.*, 2020] T. Yu, R. Wang, J. Yan, and B. Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *ICLR*, 2020.

[Zaheer *et al.*, 2017] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *NIPS*, 2017.

[Zanfir and Sminchisescu, 2018] A. Zanfir and C. Sminchisescu. Deep learning of graph matching. In *CVPR*, 2018.

[Zass and Shashua, 2008] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *CVPR*, 2008.

[Zhang *et al.*, 2015] Y. Zhang, J. Tang, and Z. Yang. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *SIG KDD*, 2015.

[Zhang *et al.*, 2019] Zhen Zhang, Yijian Xiang, Lingfei Wu, Bing Xue, and Arye Nehorai. Kergm: Kernelized graph matching. In *NIPS*, pages 3330–3341, 2019.

[Zhou and Torre, 2016] F. Zhou and F. Torre. Factorized graph matching. *IEEE PAMI*, 2016.

[Zhou *et al.*, 2015] X. Zhou, M. Zhu, and K. Daniilidis. Multi-image matching via fast alternating minimization. In *ICCV*, 2015.

[Zhou *et al.*, 2018] F. Zhou, L. Liu, K. Zhang, G. Trajcevski, J. Wu, and T. Zhong. Deeplink: A deep learning approach for user identity linkage. In *INFOCOM*, 2018.